Methods and systems for providing a secure data distribution via public networks

The present invention relates to systems having a public key infrastructure used to distribute digital data from one or more servers or content providers via a network to a plurality of users. In particular, a secure distribution and provision of digital data over a public network such as the Internet is described by the present invention.

In a common public key infrastructure and in systems using public key encryption schemes for a secure data distribution over potentially insecure distribution media, digital data is specifically encrypted for each recipient. In order to provide a verifiable integrity and authenticity of the encrypted and distributed data, one has to rely on keys, in particular on a public key pair, which are issued or certified by a trusted third party such as a Certification Authority (CA). This reliance on a third party poses a disadvantage to the security of the data distribution system. The security is established by the secrecy of the corresponding private key, which is usually known to the trusted third party or Certification Authority. This represents a single point of failure for the secure data distribution system. In addition, the service provided by a trusted third party or Certification Authority usually has pending fees and royalties that have to be paid by clients or service providers. It is, therefore, desirable to have a system capable of establishing a secure data distribution in a system having a large number of parties that communicate via a public network, which is on one hand effective in terms of security requirements and computational cost, and on the other hand does not rely on a third party. In addition, common data distribution systems providing a high level of security concerning data integrity are usually computationally expensive, especially when dealing with a large number of recipients. It is, therefore, the object of a first aspect of the present invention to provide a method and system providing for participating servers and user terminals a secure data distribution that reduces effectively the computational cost and the amount of data to be provided to each recipient while being independent of a trusted third party. Nevertheless, the system should support the use of conventional public key encryption schemes using trusted third parties. Additionally, there is a need to provide such systems and methods that are adaptable and flexible in handling different kinds of data or different parties in the system in a different manner,

especially when dealing with a large number of users and digital data to be distributed. According to one aspect of the present invention, there is therefore disclosed a method and system for securely providing digital data to a plurality of client terminals by a server using lists of fingerprints for the distributed digital data that provide for a verifiable integrity and authenticity of the data, as well as providing these lists of fingerprints or parts thereof effectively and with low transmission and computational cost to the clients using further fingerprints in a similar manner.

Another aspect of the present invention is dedicated to the distribution of digital data, while at the same time providing for an effective and easy to implement concept of controlling the access of that distributed digital data after it has been stored at the recipient side. Usually, digital data that is provided via public networks and/or that is accessible by a large number of parties in the system must not only be securely and confidentially distributed to several users in a system by asymmetrically encrypting the data, but the secrecy has to be guaranteed for at least a certain time period after the data has been received at the recipient side. In a common data distribution system, this has to be accomplished by deleting all previously securely distributed data after that certain time period as required; for instance, controlled or enforced by the provider of said digital data. So far, this problem is independently solved from the secure and confidential data distribution process. In particular this is accomplished by physically deleting all received data as well as all possible copies thereof. It is, therefore, desirable to design a system and method that can combine the secure data distribution with an effective and easy to implement way of controlling the access to the encrypted data selectively for different time periods and/or selectively for different users. Thus, there are disclosed methods for automatically revoking or deleting public or private keys used in a public key encryption scheme and/or a public key signature scheme used for a secure data distribution process.

A further aspect of the present invention is dedicated to a secure data distribution in a public key system providing for several parties to jointly or successively encrypt and decrypt or to sign and verify the distributed digital data. In an asymmetric encryption schemes, only a symmetric key is asymmetrically encrypted, whereas the data itself is symmetrically encrypted using the symmetric key. When applying

different layers of encryption to this data corresponding, for instance, to different parties in the system, the entire data has to be again encrypted and accordingly decrypted. This means high computational cost along the data distribution process. Since similar underlying processes of the encryption schemes are also applied to digital signatures, the same disadvantage applies to digital signature schemes wherein different entities or parties have to sign and verify digital data. It is, therefore, desirable to present a method and system providing for an easy and effective layered encryption and/or layered digital signature functionality in terms of computational cost. The present invention therefore describes methods providing for an effective and low complex layered encryption as well as layered digital signatures within a data distribution system.

Yet another aspect of the digital data distribution system according to the present invention is directed towards a specification of the path of the distributed digital data via a public network. It is desirable for a secure digital data distribution system and method to provide a possibility to specify and control the distribution path of the data with regard to network nodes the digital data has to pass on its way to a final recipient. This subject is not addressed sufficiently by the art, especially in the light of complexity, minimal transmission overhead and computational cost. The present invention, therefore, describes methods comprising method steps addressing this issue using asymmetric encryption and/or digital signature schemes respectively.

Typically, asymmetric encryption using a public key infrastructure employs symmetrically encrypting digital data while asymmetrically encrypting the symmetric key information. This symmetric key information is a shared secret to the sender and recipient of the distributed digital data. Usually, this shared secret is established or provided whereby requiring a trusted third party which either is directly involved in the establishing process for the shared secret or has to certify information along this process. This reliance on a trusted third party is, as already pointed out, a disadvantage of those systems. It is, therefore, desirable to include in a data distribution system a low complex and effective possibility to establish a shared secret which can serve as a symmetric key information between two parties of the system. Further, in a common system having a large number of users, a server has to store and maintain a record of all participating users and their

respective shared secrets. This, in particular, poses a disadvantage for the server in terms of storage complexity, computational cost and computing time. It is, therefore, desirable to have a secure data distribution system and method providing for the capability of establishing a shared secret with a plurality of user terminals with a server without a need to store, update or re-obtain these shared secrets for each participating user by the server. The present invention, therefore, describes a method and system using hash values computed on randomly generated tokens to establish a shared secret between a server and a client terminal that can server as symmetric key information without the requirement to store this shared secret by the server for each user.

It is the object of the present invention to provide systems and methods, as well as computer programs that accordingly control a plurality of user terminals and servers of such systems, accomplishing and allowing for the aforementioned advantageous aspects and features. This object is solved by the subject matters of the independent claims. Preferred embodiments are defined by the subject matters of the dependent claims.

In the following the invention is described with reference to the figures illustrating:

Fig. 1 a public key infrastructure system;

Fig. 2 functional units of a hash value server or certification authority server;

Fig. 3 a table storing a list of hash values together with unique identifiers;

Fig. 4 table storing a list of public keys, associated user ID and certificates;

Fig. 5 a flowchart of an administration process for a hash value list;

Fig. 6 a first part of a flowchart for a public key authentication process;

Fig. 7 a second part of the process of Fig. 6;

Fig. 8 an exemplary illustration of auxiliary hash values and lists thereof;

Fig. 9 a system comprising a network having several network nodes;

Figs. 10 to 12 a high level flow chart illustrating the basic functionality and flow for a layered encryption of distributed data involving several network nodes of the network; and

Figs. 13a and 13b a high level flow chart illustrating the basic functionality and flow for establishing a secure and encrypted communication for the secure distribution of digital data;

The present invention is described with respect to several aspects of the secured data distribution system under reference to accompanying drawings and/or several embodiments in each matter. The different most apparent aspects and their embodiments are outlined to provide a higher intelligibility in this matter but are not meant to limit the scope of the present invention. In fact, the described aspects and embodiments can be combined without departing from the teachings of the invention. Those skilled in the art will appreciate that any technically meaningful combination of all embodiments without a limitation to a specific aspect of the present invention can be combined and the scope of the .

In the following, a first aspect of the invention is described with respect to hash values used as fingerprints for the distribution of digital data and in particular for the distribution of respective public keys used in the public key system.

Following scenarios and means are meant to be exemplary rather than exhaustive. In particular, should be noted that the following aspects of the present invention are not limited to public keys, but apply to any kind of digital data, such as program files, data files, configuration files, software code, a new version or update of any afore mentioned, or combinations thereof. Because the currently preferred embodiment mainly uses this system to distribute public keys and to provide maximum intelligibility by choosing one specific kind of data, most parts of the following description refer exclusively to public keys as the digital data to be securely distributed.

An exemplary public key system according to a first aspect of the present invention is illustrated in Figure 1. The system comprises first and second certification

authority servers 11 and 13, a hash value server 12 as well as client terminals 14 to 17. The servers 11 to 13 and the clients 14 to 17 are connected to a public network such as the Internet. The certification authority server 11 and 13 account for the common use of one or more such certification authorities in public key systems having a public key infrastructures. According to the present invention, a use of certification authorities within the system is supported, but not required for establishing a secure provision of digital data, e.g. public keys, as will be described subsequently. Furthermore, a hash value server or a certification authority may as well be emulated by a distributed peer-to-peer system, for example formed by the client terminals 14 to 17 or subsets thereof. An associated public key PK1 to PK4 exists for each of the client terminals 14 to 17 or the corresponding users of the client terminals 14 to 17.

In general, client terminals 14 to 17, i.e. on behalf of the respective users, and servers 11 to 13 communicate with each another, via the public network. A direct connection, preferably between each of the servers 11 to 13, which is not illustrated in Figure 1, may provide a more secure communication path than the Internet if required.

The commonly used scenario of performing a secure communication between two terminals within a public key system that provides a certain level of verifiable data integrity and user authentication can be resumed as follows. Upon request of the owner (i.e. client) of public key PK1, a certification authority, e.g. CA1 11, initially issues a certificate for public key PK1 of client terminal 14, thereby certifying that PK1 is authentic and associated with client terminal 14 or its respective user. This certificate cert_CA1 (PK1) is publicly available or provided to the remaining client terminals 15 to 17 in the system. Client terminal 14 can then receive data that is encrypted with PK1 by these terminals. Such an authentication of public key PK1 however assumes that CA1 is a trusted third party, which has to be trusted by all terminals performing such a communication. It is further required to authenticate the PK of CA1 by means of a certificate chain to a root CA. In general, the concept of a certification authority is based on an unforgeable certificate, which can be verified using publicly known information, but has to be issued using private information only known to the certification authority itself. This shows one more consequence. Not

only has the certification authority to be trusted, but also has the certification process be protected against substitution, in other words it must be ensured that the information for verifying a certificate, i.e. the public key of CA1, cannot be replaced by an adversary.

The hash value server 12 stores a list of hash values for public keys. According to the currently preferred embodiment, these public keys are additionally signed by at least one of the certification authorities 11 or 13. A hash value of the public key PK1 is stored in the hash value list of the hash value server 12. As described in more detail below with reference to Figures 5, the hash value server 12 calculates a hash value for a stored list of hash values and provides the calculated hash value and the list of hash values so that they are publicly available for all terminals. In the following, the hash value of the list of hash values is also referred to as a meta hash value.

The information stored in the hash value server 12 may be provided for public access or at least accessible for dedicated client terminals of the system. In particular, providing the information also includes forwarding it either upon request or automatically to a list of predefined client terminals. Different embodiments of how the hash values list and the meta hash value are provided and distributed to the client terminals are discussed following the subsequent description of the general concept.

In one embodiment of the invention, client terminal 15 receives the list of hash values and the meta hash value thereof from the hash value server 12. Based on the received meta hash value, the client terminal 15 performs an authentication or verification process for the public key PK1 of the client terminal 14 before using the public key PK1 for verifying, authenticating or encrypting data. Moreover, the client terminal 15 may as well check the authenticity of its own public key PK2 included in the list of hash values. A corresponding process in a client terminal is described in more detail below with reference to Fig. 6 and 7.

The hash value server 12 of Fig. 1 may as well be implemented as a part of the certification authority servers 11 and 13, as part of another server in the network storing or managing digital data to be securely distributed, or by means of a peer-to-

peer system of the client terminals. The used hash algorithms may for example be SHA1 or MD5.

Figure 2 illustrates functional units of the hash value server of Figure 1.

The typical hash value server comprises a CPU21, a network interface unit 22, connected to the Internet, operator I/O units 23 for interacting with an operator, storage means 24, as well as further storage means 25, 26.

The operator I/O units 23 particularly comprise monitor, mouse and keyboard. Furthermore, the network interface unit 22 allows the server to receive requests for information from the client terminals, to transmit the stored information or to receive input information. Input information may for example be received from the certification authority servers for further data, for example public keys, to be added to the list of hash values. Particularly in this regard, a not illustrated direct interface unit may provide a secure direct connection to at least one of the CA servers.

The storage means 24 may be formed by RAM, EEPROM, ROM, a hard disk, a magnetical disk drive and/or an optical disk drive. An operative system of the server as well as application software to perform the required operations is stored in storage means 24.

In this example, the further storage means 25, 26 are formed by a first storage unit 25 for storing hash values and a second storage unit 26 for storing public keys as well as certificates thereof. In general, the storage means 26 can hold any data, for example, partitioned in one or more lists of data to be securely distributed. The storage unit 25 holds a list of hash values for the public keys or data stored in storage unit 26, as well as the meta hash value for the list of hash values. This storage unit 25 may further store a temporary list of received hash values separately stored from the list of hash values currently provided to the public.

Figure 3 illustrates a exemplary list of hash values 32 for public keys (PK1 to PK4) as stored in the hash value server. A unique identifier associated to the public key and thus also associated to the hash value thereof is correspondingly stored in column 31. Unique identifiers are preferably formed by e-mail addresses of the respective owners of the public keys PK1 to PK4. The list of hash values 32 further

stores a meta hash value for a list of hash values of the certification authority CA2. The list may further comprise a hash value for the public key of the certification authority CA2.

Finally, a meta hash value is calculated for the list of hash values 32 or preferably for the list of hash values 32 and associated e-mail addresses 31.

Figure 4 illustrates data, in this exemplary case public key/certificate pairs, as they may be stored in the storage unit 26 of Figure 2.

Column 41 comprises a user ID as a unique identifier for a user. The user ID may for example replace or correspond to the e-mail addresses of Figure 3 or may even be mapped thereto in a further reference table. This unique identifier 41 may also be a social security or any other identification number. Preferably, the hash value server or the certification authority ensures that there is always only one valid public key for each unique identifier. This unique information may also enable the sender of a message that is encrypted using said public key to identify the owner of the public key. Column 42 comprises a list of public keys for the users identified in column 41. Column 43 comprises a list of certificates for the associated public keys, the certificates being issued by one of the certification authorities CA1 or CA2. Each entry in the list 41-43 corresponds to one entry in the list of hash values of Fig. 3.

Besides public keys PK1 to PK4 of the users 1 to 4, the last item of the table in Figure 4 comprises a public key PK_CA2 of the certification authority CA2. A corresponding certificate CA1_cert (PK_CA2) is issued by the certification authority CA1.

Furthermore, the tables illustrated in Fig. 3 and 4 may additionally comprise non-illustrated data fields such as a revocation information, indicating if a hash value or a corresponding certificate has been revoked, or update information, indicating a date or time when the hash value has been updated.

A process of administering a list of hash values in a hash value server is now described with reference to Figure 5 according to one embodiment of this first aspect of present invention. The described process refers to lists of public keys as the data to be transferred to and authenticated by the client terminals of the system. This and all following scenarios are meant to be exemplary rather than exhaustive. In particular should be noted that the present invention is not limited to public keys, but applies to any kind of digital data, such as program files, data files, configuration files, or combinations thereof.

Initially, in a step 52 a public key PK is received which may be signed by a certification authority CA. For instance, the PK may be part of a certificate issued by the CA.

A hash value of the public is calculated in step 53.

Subsequently, the signature of the certification authority CA may be checked in an optional step 54 in order to verify that the public key is actually signed by and/or received from the certification authority. Such a verification step is achieved by applying the public key of the CA to the existing, received CA's signature according the employed public key signature or certification process. In case the signature cannot be verified, the process is terminated.

In step 55 the calculated hash value is added to a list of hash values, which is stored in the hash value server. For the supplemented list of hash values a meta hash value is calculated in step 56. The list of hash values may be signed by the hash value server in step 57. Finally, the hash value list, the meta hash value thereof and optionally the signature of the hash value list is provided in step 58.

The step of providing 58 may for example be implemented by storing the information in the hash value server and transmitting the same upon request, forwarding the information to a list of predefined destinations or forwarding the same to one or more predefined publication means.

Preferably, in the step of adding 55 the calculated hash value is initially added to a temporary list of hash values stored separately from the list of hash values currently provided to the public. Furthermore, a time interval may be defined for performing

the steps 56 to 58 for example daily, weekly or monthly only. Hence, new hash values received within the given time interval will be intermediately stored in the temporary list for being added to the published list after expiration of the time interval. Moreover, In order to inform about the relevance of the meta hash value, the time or date of the calculation of the meta hash value may be stored and provided together with the meta hash value.

Furthermore, a unique identifier such as the email address of the public key's owner may be also received from the CA, assigned in the hash server, or approved by at least one of these servers, according to the requirements of the public key system.

Figures 6 and 7 illustrate an authentication process as performed in a client terminal according to one embodiment of this first aspect of the present invention. As with Figure 5, the underlying authentication process of Figures 6 and 7 is illustrated for public keys, but applies to digital data in general.

In the process 60 to 68 of Figure 6, a hash value list and a meta hash value thereof are initially received in step 61 from a first hash value server. Thereafter, a second meta value for the hash value list of the hash value server is received in step 62 from a second hash value server. Finally, in a step 63 of comparing the received meta hash values, it is determined whether both meta hash values correspond to each other.

Furthermore, each of the steps 61 and 62 may additionally comprise a step of verifying a signature issued by the corresponding hash value server for the meta hash value and/or a step of decrypting the received information, if the received information is encrypted by means of a key for example derived from a mutual authentication process. As it will become more apparent in the following, various sub processes, for example steps 64 and 65 or steps 66 and 67, may optionally be combined to the general steps 61 to 63 in order to modify a required level of security in the authentication process for the public key.

The process 60 to 74 may be terminated, if in any step of comparing, verifying or authenticating indicates a possibly faked key which accordingly should not be used for subsequent communication. For example, if the comparison result of step 63 indicates deviating hash values, the list of hash values can not be considered as a

trusted list. However, after such a single negative authentication result the process does not have to be terminated, but may as well be continued with an alternative or additional sub process for authenticating the public key in question. In particular the steps of receiving and comparing 61 and 63 or 62 and 63 may for example be repeated using yet another source to obtain the data. It is again noted that a hash value server may as well be formed by a certification authority server or a peer-to-peer system of clients emulating the same.

In addition, the number of meta hash values obtained for one hash value list that are compared against each other, is not limited to initially two as indicated by this exemplary process shown by Figure 6. In fact, the more meta hash values are obtained from ideally independent sources the higher the level confidence that the hash value list is authentic and thus the integrity of the covered data is given, given a successful comparison. Therefore, in another embodiment, a certain number of meta hash values from different sources for one hash value list may be required to establish a certain security or confidence level. This required level may vary for different kinds of data, e.g. it may be higher for public keys than for files containing less sensitive information such as music files. It may also vary based upon user instructions or requirements set by a content provider or distributor.

After the step of comparing 63, the meta hash value is calculated 64 by the client terminal based upon the hash value list received from the hash value server. The calculated meta hash value is compared in step 65 with one of the received meta hash values.

Moreover, in step 66 a hash value is calculated for a specific public key PK covered by the hash value list. Preferably this step 66 and the succeeding step 76 are carried for at least the desired public key to be authenticated by the client terminal. Based on the calculated hash value H(PK), the corresponding hash value stored in the received list of hash values can be verified by comparison 67. If such a match can be verified, the consistency of the public key and the hash value list is established. Thus by verifying the integrity and authenticity of the hash value list using the meta hash value, the integrity of the public key covered therein can be authenticated as further described subsequently.

The process illustrated in Figure 6 is continued with steps 71 to 74 of Figure 7, comprising 3 further sub processes of the authentication process.

An optional step 71 of receiving a user input for verifying the hash value is provided, for cases which can not be handled automatically such as a hash value received by a user via e-mail. The step of receiving 71 may comprise requesting the user input, receiving same and evaluating correspondence to the calculated hash value of the public key. A similar process may be performed for the meta hash value of the list of hash values.

Furthermore, a certificate of the public key in question may be verified in an optional step 72. Finally, in order to check whether the third party or alleged owner of the public key in fact holds the private key corresponding to the public key, a signature of the private key, typically applied to random data provided by the client terminal as a random challenge, is verified in a further optional step 73. In general, step 73 may accomplish any available Zero-Knowledge-Proof-of-Knowledge method to verify the possession of the corresponding private key to the public key of the alleged key pair owner.

According to the currently preferred embodiment, the compilation, computation and distribution of the hash value lists and their respective meta hash values is performed as follows.

As described by aforementioned processes, a hash value is computed for every entry in a physical or virtual list of data. For example, a hash value is computed for each public key or, where applicable, for each public key together with its respective certificate. These hash values, each associated with at least one unique identifier, form the hash value list. The meta hash value is computed from this complete hash value list. Thus, each time a new valid meta hash value is effective for the continued list of data as accumulated by the time, also the complete hash value list has to be obtained by a terminal for the above-described authentication process. This might for instance be required on a daily, weekly or monthly basis.

Because this hash value list can be significantly large and a client terminal (or the associated user) may only want to authenticate a single or a few entries of this list,

e.g. a specific public key, it is desirable not having to distribute the complete hash value list each time a specific data entry has to authenticated.

The preferred embodiment concerning this matter, therefore provides for the use of auxiliary hash values and a respective list thereof, as described subsequently in further details with reference to an exemplary illustration in Figure 8.

Referring to Figure 8, list 800 comprises digital data 801 that is to be distributed securely among the client terminals. This digital data 800 may for example correspond to or represent the public keys 42 or the public keys 42 together with the respective certificates 43 as shown in Figure 4. Each data entry in the list 800 has a unique identifier 802, similar column 31 and column 41 in Figures 3 and 4. The list 800 further comprises a hash value 804 computed for each data entry 803, correspondingly the hash value 32 in Figure 3.

Furthermore, a second identifier 801 is associated with each entry in list 800, preferably specifying the time when the data has been listed in ascending order as the list continues. In the following, column 801 is referred to as the entry time of the associated data. It should be noted that this time might also correspond to the computation time of the respective hash value 804 if that differs from the entry time of the data itself. Furthermore, only one unique identifier might replace both, the associated time in column 801 and the unique data identifier 802, for each data entry. Another scenario is to use one of the lists illustrated in Figures 3 and 4 and provide the timing specification in a further list, which respective entries are linked to each other preferably via pointers. It is however also possible not to use such identifiers 801 but instead realize the subsequently described concept using other methods or means that provide the same functionality of linking different auxiliary hash values 812 to their corresponding segments in table 800.

The entries in list 800 are divided into consecutive segments. Preferably, these segments are strictly consecutive and non-overlapping. They can however also overlap or even be interleaved. The currently preferred embodiment divides these segments according to a timed schedule. Naturally, new entries of data, e.g. new issued public keys for possibly new clients and/or users participating in the network or new released software products, are added to the list 800 over time. These

entries are accumulated and the hash values 804 are then computed if necessary. After a certain time period, for instance hourly, daily or weekly at a predetermined time, all latest entries not already covered by another segment form the next segment. As illustrated in Figure 8, at times T150 and T200 a segment was completed. Consequently all entries enlisted after time T150 up until scheduled time T200 form a segment, namely all data entries D101 to D150. Another scenario is to segment list 800 according to the number of data entries, thus forming segments of equal numbers of entries rather than of equal time spans for data entering the list. Yet another scenario is to use different segments for different kinds of data, or different kinds of data identifiers such as user email addresses. The latter scenario represents one segmentation procedure not only scheduled by the entry time into the list.

Each segment of list 800 is used to compute an auxiliary hash value. This is analog to the generation of the meta hash value for the lists illustrated in Figures 3 and 4, when referring to each segment as a separate list. Similar to the meta hash value, the identified hash algorithm for the auxiliary hash value can be applied to either exclusively the considered data or to the data and the remaining list entries in each list row. In the present case of the auxiliary hash value, the hash algorithm can be applied to all hash values 804 in the respective segment, or preferably to all hash values 804 and the respective data 803, data IDs 802 and times 801 in the segment, or any subset or predetermined combination thereof. The auxiliary hash value 812 for each segment of list 800 are entries in another list 810, which further comprises an hash identifier 811 for each auxiliary hash value. This list 810 can be stored as a separate list or as a "virtual list" if for example included in list 800 with an additional indicator such as a flag identifying the auxiliary hash value entries.

The identifier 811 preferably represents a time, and in particular the latest time in each associated segment of list 800. For example, the third segment illustrated in Figure 8 comprises all entries in the time span T151 to T200. Since the segments are mutually exclusive but adjoined, the latest time in each segment unambiguously identifies each segment when using a scheduled segmentation approach such as an hourly completion of the respective segment comprising the latest entries in list 800. Thus, the currently preferred embodiment uses this latest time as the hash ID

811 in. In another embodiment however, this mapping from auxiliary hash value 812 to the respective segment can be established by any other available unambiguous linking method, means or concept. Note, that once the auxiliary hash value has been computed and stored, this mapping is not necessarily required to be unambiguous in both directions, as it will become apparent subsequently.

The entries in list 810 are now used to compute the meta hash 821, illustrated in a separate list 820. Again, this list could be included in any other list, for instance in list 800 or 810. One might even distribute or publish the latest meta hash value and delete the old one, thus not storing the meta hash values as indicated by list 822 in Figure 8. This meta hash value is computed as already described above with references to Figures 3 to 5. Apparently, list 810 can be treated as the list shown in Figure 3, wherein each auxiliary hash value 812 corresponds to a hash value in column 32 and hash IDs 811 correspond to column 31. Like in the authentication process already described above, the meta hash value is distributed and associated with a timing or validity information, for instance a hash ID 822 specifying its computation time, as required to indicate whether a received meta hash value and the corresponding hash value list are still valid or for instance whether there exist already a later version. In principle, this information is only required for one of the meta hash value or the correspondent hash value list HVL, as long both are identifiable as corresponding to each other.

According to the currently preferred embodiment, the meta hash value 821 is computed along with the latest auxiliary hash value 812 covered by the meta hash value. Referring to the example shown in Figure 8, meta hash value mh1 is calculated at time T150 as indicated by its respective hash ID 822. This time identifier is the same as the hash ID 811 of the respectively latest covered auxiliary hash value ah2. Another embodiment may apply independent schedules for computing the hash values 812 and 821 that do not match with respect to completely covered segments. As indicated in Figure 8, meta hash value mh2 is computed at time T222, whereas the latest covered auxiliary hash value ah3 covers data entries in list 800 up to time T200. This scenario illustrates in contrast to the currently preferred embodiment the possible case according to which a meta hash

value (i.e. mh2) does not cover all data 803 (i.e. D201 to D222) enlisted at the time (i.e. T222) the meta hash value has been computed.

The authentication process according to this currently preferred embodiment differs from the above-described authentication process as follows. Steps 61 to 65 are applicable as described with reference to Figure 6. These steps include the step 63 of comparing at least two meta hash values for the same currently considered hash value list, each obtained from different sources in steps 61 and 62. Further in step 65, these meta hash values are compared versus a local meta hash value, computed by the client terminal from the respective hash value list, which is in this embodiment list 810. By verifying that all considered meta hash values for this hash value list are the same, the authenticity and integrity of this hash value list can be established, based on the diversity of different and fairly independent sources of the obtained meta hash values. The meta hash value 821 is then computed from the auxiliary hash values 812. Thus, the data 803 to be authenticated by the client terminal, for example a public key PK, does not correspond directly to any hash value in the auxiliary hash value list 810 as received in step 61. Therefore, before proceeding with step 66 and succeeding steps, the following steps have to be performed.

Only to provide maximum intelligibility, the following sections refer exclusively to public keys. These public keys represent the digital data 803 as one specific usage case. In addition, Figures 6 and 7 illustrating a more general embodiment of this process do also refer to this specific usage case wherein public keys symbolize the digital data to be authenticated.

According to a first embodiment concerning the authentication process, the client terminal establishes which auxiliary hash value 812 corresponds to the currently considered public key 803 that has to be authenticated and which segment of the list 800 corresponds to this auxiliary hash value 812. Thus the client terminal establishes which segment comprises said public key. The client terminal then requests this identified segment of list 800 from at least one predetermined server, preferably from the same hash value server also providing the hash value list. To achieve these establishing and requesting steps, different approaches are applicable and considered for this invention. One possible approach is to associate

each public key with time information that a verifier such as a client terminal can establish. This time information unambiguously specifies the corresponding segment in the list 800 as employed by the currently preferred embodiment. Different approaches however may not (only) use time-scheduled segments or may not use time identifiers 800, 811, and/or 822. Another approach may for instance provide an identifier associated with each key directly specifying or pointing to the respective segment. In a further approach, the hash ID 811 may comprise information providing for identification of all covered public keys 803. Thus, a client can identify the respective auxiliary hash value 812 of said public key, which in return specifies the desired segment of list 800.

A second embodiment concerning the issue of obtaining the corresponding segment of the list 800 to a currently considered public key is the following. The client terminal requests the required segment at a server by sending, pointing to or specifying the public key, preferably at said first hash value server. This server then establishes the corresponding segment and may then provide this segment directly to the requesting client terminal or at least provide information enabling the client terminal to obtain the requested segment. A combination with the first embodiment is possible, for example, the client determining the respective auxiliary hash value 812 of the public key 803 and the server determining the respective segment of this auxiliary hash value 812.

According to a third embodiment concerning this requesting of the required segment, the segment obtained according to either one of both previously described embodiments can be further encrypted and/or certified, for example by the hash value server. This implies a decryption and/or verification process performed by the client terminal after having received the segment.

According to a fourth embodiment concerning this requesting of the required segment, all segments are publicly available and accessible for each client terminal, for example published on a server accessible via Internet. Each terminal establishes which segment is required, e.g. as indicated above, and obtains the same from this server without interacting with a dedicated hash value server or certification server.

Having obtained the segment of list 800 corresponding to the considered public key 803, the client terminal can proceed with step 66 and the succeeding steps of the authentication process as illustrated in Figure 6.

An advantage of this currently preferred embodiment using the auxiliary hash values is the reduction of hash values to be distributed. When a new meta hash value is computed and provided to the client terminals, the new, augmented hash value list from which the meta hash value is computed has to be distributed as well for two obvious reasons. First, for the sake of maximum security, the hash value list has to cover the complete list of data 803 to be securely distributed or provided. Second, according to the steps 64 and 65 of the described authentication process, the meta hash value is also computed by the client terminal, which requires the complete hash value list. The auxiliary hash value list 810 from which the meta hash value 821 is computed still covers all data 803, but has fewer entries than the original list 800. Thus, a list with fewer entries has to be provided initially to the client terminals. The reduction in size from the original list 800 to the auxiliary hash value list 810 depends directly on the sizes of the segments. However, increasing the sizes of the segments implies having for instance a larger time frame for each segment, which might be undesirable. It further implies that a larger segment and thus more data has to be transferred to the client terminal within the authentication process, which also appears undesirable. For this, yet another embodiment provides for staged segments applying the concept of the auxiliary hash value list successively again to the auxiliary hash value list 810, according to a tree structure having the meta hash value as the root. In particular, the obtained hash value list 810 is segmented according to any of the above-described embodiments. These segments are used to compute a second set of auxiliary hash values similar to list 810 instead of the finally desired meta hash value 822. This procedure can be repeated for any number of stages, where the stage depths may be predetermined or variable. In particular when using the above second embodiment according to which a dedicated server determines and provides the requested segment to the client terminal within the authentication process, the successive staging and generating of auxiliary hash values would be entirely internal to the server requiring a list management and linking functionality not affecting the client terminals. Thus the server or several servers can create staged segments as appropriate for their internal functionalities.

Furthermore, segments at different stages as well as within one stage may vary in size. Different auxiliary hash values of different segments and stages may even be combined in a new segment. In particular, only a subset of the hash values in the auxiliary hash value list of one stage can be further segmented and "replaced" by an auxiliary hash value of the next stage.

Another embodiment concerning the provision of the hash value list being required to calculate the meta hash value, is exemplarily outlined as follows. This complete hash value list, either list 800, 32 or if applicable list 810, is required to compute the meta hash value. It should again be noted, that the complete list in this sense refers to at least all hash values 804, 812, or 32 of said list, but may contain an arbitrarily but predetermined number of further entries of the respective list. However, it can be taken advantage of the fact that this list is accumulated over time by adding new entries to this continued list. Once enlisted in the list, an entry remains unchanged. Thus, a new hash value list corresponding to a new meta hash value comprises all entries of the previously effective hash value list. This allows to update a hash value list rather than to replace the entire list in a client terminal when a receiving a new meta hash value. Assuming the client terminal is already in possession of a hash value list when receiving a new meta hash value for this list. Instead of obtaining the respective new hash value list the client terminal can only obtain the new entries to this hash value list and then amend the already locally stored list appropriately. This embodiment can be combined with the segmented list approach with possibly further staged segments.

In all embodiments described in connection with the hash value lists and the meta hash value, different hash values can be cross posted between different lists and/or segments. A currently effective or an obsolete out-of-date meta hash value of one original list 800 can be added to the same list 800, to any associated auxiliary list 810, or to any other list 800 or 810 not associated with said original list 800. This cross posting of hash values, preferably of meta hash values, increases the distribution diversity of the meta hash value and hence the confidence and security level when authenticating the integrity of the meta hash value and its associated lists. This cross posting of typically one or a few hash values does not significantly increase the amount of data to be handled by the system, and thus effectively

increases the security level of the system without affecting its performance. This approach is especially advantageous for a small number of available different sources from which a client terminal can obtain the meta hash values used in the authentication process.

In addition to cross posting single hash values between different lists, hash value lists, for example originating from different hash value servers 12 or certification authorities 11, 13 may as well be combined in a single list. A third party may additionally provide such a combined list.

Another aspect of the present invention refers to the distribution and provision of the meta hash value as required for the authentication process. The currently preferred embodiment for this distribution of the meta hash value is the following.

Instead of receiving the meta hash value on demand, e.g. together with its respective hash value list, as indicated above, the meta hash value is sent to each client terminal possibly multiple times and not only in connection with the actual authentication process. The currently preferred embodiment attaches the effective, currently used meta hash value to messages sent to and/or between client terminals. Because this is a single hash value, the communication of the terminals is not overly affected by attaching it to a message sent to the terminal. This may be done once for each new meta hash value, thus following for example a daily or weekly schedule for generating a new meta hash value. However, the preferred alternative is to attach the meta hash value to several messages, for instance regularly and/or mandatory as part of an underlying communication protocol. This attaching to regular messages received by the client terminals can be arranged or caused by at least one server, such as a hash value server. In addition client terminals itself may attach a previously received meta hash value to messages sent by the terminal. For the latter case, it can be required that according to some given security specifications the meta hash value is attached only if it has been authenticated by the terminal, for example with a certain confidence level achieved after a successful comparison of a minimum number of previously received respective meta hash values from different sources.

This attached meta hash value can additionally be signed by the entity that attaches it to a message, i.e. a server or a terminal. This ensures – within the security limits of the underlying digital signature scheme – that indeed the sender of the message and no adversary having intercepting the message has attached the meta hash value. Accordingly, after having received a message and detached the meta hash value, the associated signature must be verified by the terminal before proceeding further.

This distribution scenario without an explicit request of the meta hash value by the receiving terminal at the time of its use requires that the terminal can establish the validity of the meta hash value. In other words, the client terminal must be capable to determine whether the received meta hash value is indeed the latest published meta hash value as required in the authentication process. One possible way is to associate time information to the distributed meta hash value. This allows a terminal to establish for example whether a meta hash value is meant to correspond to the latest received hash value list. Preferably, this information comprises the expiration and/or creation time of the meta hash value and/or the corresponding hash value list. Thus, its validity can be established possibly by further taking into account a known schedule for generating the meta values. Another scenario for establishing the validity of the meta hash value is to assign a unique identifier to each value that has to correspond to the hash value list received upon request and therefore known to be up to date, whereby said hash value list has preferably also an identifier.

This embodiment allows a client terminal to compare many meta hash values received from different sources, e.g. terminal. Thus, the confidence level of the established authenticity by comparing step 63 and by comparing them with the locally computed value in step 65 is increased. This holds in particular, if only previously authenticated meta hash values are further distributed to other terminals. This distribution diversity of the meta hash value effectively further decreases the chances of an undetected single point of failure in the system, e.g. caused by a malicious adversary.

Another aspect of the present invention is dedicated to a secure distribution of digitally encoded data when this data is stored and/or accessible prior or after its actual provision to the destined recipient.

In order to provide maximum flexibility in terms of usage models that are supported by the distribution model according the present invention, the secure distribution of digital data also provides for an easy solution to ensure data security in case the digital data is stored and – possibly publicly - accessible prior, after or during the actual data provision and transmission process. In addition, in a common data distribution scenario, digital data is often transferred via public networks and many third parties and clients may have access to the distributed data. Therefore, secured distribution has to account not only for the actual data transmission transactions but also for a preceding and/or succeeding access to that distributed data. Thus, it is desirable to provide an easy approach for controlling the access to the distributed data in connection with the distribution process. One apparent solution is, for instance, a controlled data access mechanism that is based on a timed schedule such as a given time span defined by the actual data transmission transaction in which the data can be accessed. This ensures that after the time span has expired and data access capability has lapsed, the security and confidentiality of that data can be guaranteed. Naturally, this scenario also occurs and is applicable, if certain data has to be deleted at a given time or after a predetermined time period. Applying this distribution model, the deletion can be ensured without having to control the physical deletion of that data. Moreover, the potential risk that stored data or automatically-generated log information and back-ups of distributed data, for instance stored on a server, becomes disclosed to the public can be minimized by the following currently-preferred embodiment of this aspect of the present invention.

According to this aspect of the present invention, applying a public/private key pair in an appropriate public key encryption scheme asymmetrically encrypts all digital data distributed through this distribution system. For this, all data is – completely or partially – encrypted with a public key. All intended recipients and parties authorized to access the data in plain text are in possession of the corresponding private key required for the respective decryption process. Consequently, only those parties are able to decrypt the encrypted data and gain access to the distributed data. The system now controls the usage of different key pairs according to predetermined schemes. Revoking and/or deleting individual keys establish different functionalities and features of the secured distribution system as described by the following embodiments regarding this matter.

According to one embodiment, the distribution model provides all recipients and clients authorized to access the data with the respective private key required to decrypt the distributed and encrypted data. In this and all subsequently described embodiments, the provision of the private key is accomplished by a secure key distribution transaction. Because many key distribution and key management systems are widely used and known, whereby some involve secured and trusted communication media, and because this key distribution can be seen as a generic and independent preliminary transaction in this specific instance, this key distribution will not be considered in further detail at this point. It is assumed that an appropriate means to distribute said private key is available. Preferably, these keys are securely distributed using the described hash value lists and meta hash values.

The system further causes this private key to be deleted or revoked in compliance with this embodiment of the present invention. This implies that following this controlled revocation or deletion of the active private key, all authorized clients previously in possession of that private key cannot gain access the encrypted data as plain text anymore. Consequently, all data previously encrypted by using this key pair is effectively prevented from being accessed without having to delete the actual data. This is particularly advantageous for large amounts of data that has been distributed and encrypted using one specific key pair, since only the rather small key information has to be deleted or revoked. Moreover, in general, a revocation or a deletion of one or a few private keys at a client or client terminal can be controlled or accomplished easier and more effectively than a deletion of all data previously distributed to that client, including possible back-ups and copies thereof.

Once a private key has been deleted or revoked by the system, a new key pair is employed in the same manner as previously described. This fundamental concept can easily be applied to different usage models as discussed subsequently where different exemplary embodiments of the invention are provided. These embodiments are not meant to be exhaustive and the invention is not limited to those cases, in particular any combination of these embodiments and the technical teachings thereof also describe the present invention.

Instead of using only one key pair at a time, several key pairs can be employed concurrently or in overlapping time periods. In addition, different key pairs for

different kinds of data, different groups of recipients, or different providers of digital data can be used to distinctively control the distribution of digital data. The generation and/or the revocation of keys might follow a timed schedule. The system can generate a new key pair and require the sender or distributor to use the respective private key for encryption for a first time span, e.g. on a monthly or weekly basis, but revoke or delete the private key at the clients or recipients not until a second time period has expired, e.g. six months after its generation, after the encryption took place, or after the encrypted data has been transmitted to the recipient. This means that all authorized clients have a six months period to access that data that was encrypted during the first, e.g. one month, time period in which the respective public key has been used. This time-based access control can particularly be combined with the approach of using different keys for different kinds of data or users.

According to another embodiment, a key pair is generated and applied to encrypt the data prior to their distribution in the same manner as discussed for the previous embodiment. Contrary to the previous embodiment, however, the public key instead of the private key is made subject of revocation or deletion by the system. Consequently, the capability to encrypt and thus to distribute digital data is controlled by the system. This means that after the public key has been deleted or revoked, all clients in possession of the corresponding private key are still able to access all previously encrypted data. But it is ensured that no further data can be encrypted and distributed for this private key. Similar to the previously described embodiment, multiple key pairs may be used for different kinds of data, designated recipients and providers and the public keys are generated and revoked according to one or more timed schedules. This embodiment might however additionally revoke or delete as well the private key, gaining the same benefits as outlined for the previous embodiment.

According to a further embodiment of the present invention, the same concept can be employed for generating digital signatures on distributed data. For this, the data provider employs a private key and the recipients the corresponding public key of the key pair according to the underlying public key signature scheme. Revoking or deleting the private key has the effect that no further data can be signed using this

key pair. If a recipient of digital data has to verify a signature on that data in order to validate its authenticity or integrity, a valid distribution of further data can effectively be prevented by deleting or revoking the private key at the distributor or provider side of the network.

According to the present invention, any combination of the aforementioned embodiments with regard to deleting and/or revoking of keys is considered as a still further embodiment of this invention. In addition, in all of these embodiments it is assumed that this deletion and/or revocation of keys can be controlled, enforced or ensured in an appropriate manner that is sufficient for and may depend on the required security of the system. This might involve trusted devices and/or physical security of parts of the system. Additionally, all embodiments regarding the deletion and/or revocation of keys can be performed with respect to only one party of the system. In particular, this refers to the public key pair used in the respective encryption or signature scheme. This single party or client can encrypt digital data using its own public key prior to storing said data. This concept provides for secure storage functionality since only this client can decrypt the data.

Yet another aspect of the present invention refers to the secured distribution of digital data when using of several encryption schemes and/or several encryption steps. Usually in a network for distributing and transmitting digital data, asymmetric encryption schemes are employed, which are in particular public key encryption systems. Typically, in order to encrypt the data, a symmetric encryption scheme is applied to the data using a symmetric key, which is denoted S in the remainder of this description. This symmetric key S or the generation thereof often involves some randomness to increase the cryptographic security of the encryption scheme. In this case, an asymmetric encryption is accomplished by asymmetrically encrypting the symmetric key prior to its distribution.

In the following, a layered encryption scheme according to the present invention based on this asymmetric encryption concept using a symmetric key to encrypt the actual data is described in greater detail under reference to accompanying drawings that form themselves a part of this description. In a layered encryption, digital data is successively encrypted using several keys, preferably associated with different parties or entities of a network, whereby each encryption process and thus each

employed key accomplishes one of said encryption layers. In other words, the plain text data is initially encrypted using a first encryption key according to afirst encryption scheme. The encrypted outcome is then further encrypted in several stages each applying respectively another key according to the respective encryption scheme.

This successive or layered encryption usually has to re-encrypt the entire amount of data at each encryption layer, possibly under inclusion of an encryption overhead of a previous layer. According to the present invention, however, this amount of data that has to be encrypted during this layered encryption process can be significantly reduced as follows. Instead of re-encrypting the entire (after the initial encryption layer already encrypted) data, only the key information is encrypted at each layer.

When applying the concept that the data is encrypted using a symmetric key, possibly a one-time key specifically generated for this secured data transmission, not only the encrypted data but also the symmetric key is transmitted or at least provided to the destined recipient. As noted before, this symmetric key S is asymmetrically encrypted by using a public key encryption scheme. For this, the sender or distributor that performs or initiates the data encryption first applies the symmetric key S to encrypt the data respectively. Then, the sender or distributor uses the public key of the destined recipient of the data in order to encrypt the employed symmetric key S. In order to decrypt the received, encrypted data the recipient has to have access to that symmetric key S. Consequently, the recipient must first be able to decrypt this symmetric key before being able to gain access to the encrypted data. This prerequisite of being able to decrypt the symmetric key before being able to decrypt the actual encrypted data provides for the efficient layered encryption process according to this aspect of the present invention.

Instead of encrypting the entire data at each encryption layer, only the already encrypted symmetric key is encrypted. The performance gain mainly results from the fact that only the small key information has to be encrypted and accordingly decrypted for all but the first encryption layer. According to the currently preferred embodiment of the present invention, to encrypt the symmetric key at each encryption layer, a public key encryption scheme is employed. Accordingly, the destined recipient that is meant to be able to decrypt the encrypted data and,

therefore, also meant to decrypt the symmetric key used for encrypting the data, must be in possession of all private keys that correspond to the public keys used for each encryption of the symmetric key at each encryption layer.

It should also be noted that the sender who encrypts and then sends the encrypted data including the encrypted key information to the recipient may represent a plurality of different parties or entities within a network, each accounting for at least one encryption layer. That means that one entity or party of the network encrypts the data and/or the symmetric key information and sends it to another one in order to carry out all operations associated with the next encryption layer. Likewise, the recipient as referred to above may represent a plurality of recipients, each corresponding to one or more encryption layers by performing the respective decryption steps. This scenario would provide for specific dependencies between this plurality of recipients since only after the last decryption step regarding the encrypted symmetric key, the actual encrypted data can be decrypted. Therefore, the recipients that decrypt the symmetric key information according to one or more encryption layer can control access to the encrypted data for the last anticipated recipient. In case the individual decryption processes that correspond to each encryption layer have to be applied to the transmitted encrypted symmetric key information in exactly the reversed order of the corresponding encryption processes, a hierarchy can be established of parties or entities for controlling the capability of decrypting and gaining access to the transmitted encrypted data.

According to another embodiment regarding this matter of the present invention, different encryption schemes can be applied at different encryption layers. This means that different asymmetric encryption schemes can be applied at different encryption layers, for instance, RSA and Diffie-Hellman based schemes. It is, however, also considered to support deviations of the described concept that asymmetrically encrypts the symmetric key. In particular, at some encryption layers also the already symmetrically encrypted data might be again partially or completely encrypted by the sender and correspondingly decrypted at the recipient side. For this, different approaches are considered. Additionally to asymmetrically encrypting the symmetric key information, also the symmetrically encrypted data can be re-encrypted asymmetrically using the same public key. Further, only the encrypted

data might be encrypted but not the symmetric key information. The latter case implies that this specific encryption layer has no influence on whether the recipient(s) can decrypt the symmetric key. It therefore rather introduces an encryption layer that is independent of the remaining encryption layers. Also, one or more encryption layers may not apply a asymmetric public key encryption scheme but a symmetric encryption scheme using a symmetric key and encrypting this key similarly to the described initial symmetric encryption layer performed on the original plain text data.

According to the currently preferred embodiment of the present invention in this matter, the above-described layered encryption approach is applied to the distribution of digital data via a network having several network nodes. So far, the secured distribution of digital data via a network according to the present invention has been described in terms of the distributor or sender and the recipients of that digital data. The same concepts, however, can be applied to the network that is used as the underlying communication media for distributing the digital data. Figure 9 therefore shows an exemplary illustration of a network 900 having several network nodes 906-909. These network nodes are typically interconnected and able to exchange data between each other. Different terminals and servers 901-905 communicate with each other via said network 900 and can themselves be seen as network nodes. According to the currently preferred embodiment, network 900 is the Internet. The connections between the plurality of network nodes is typically established via various links between these nodes whereby one or more direct links or links via other nodes are possible. Referring to Figure 9, a sender 901 aims to send digital data to recipient 902 via the network 900. The digital data to be transmitted to user terminal 902 is encrypted by at least applying a public key encryption using the recipient's public key prior to the transmission of the in this case encrypted data. For this, the sender 901 that is considered to encrypt the data first obtains the public key of recipient 902. According to the preferred embodiment of the invention, the hash value lists as described for this public key system are used to securely provide sender 901 with the public key of recipient 902. All considerations regarding the hash value lists apply and are therefore not further discussed at this point.

Having obtained the public key of the recipient 902, the sender 901 encrypts the digital data according to the above-discussed layered encryption concept, which is applied not only with regard to the sender and recipient but also to the network nodes. According to one embodiment, one or more network nodes perform accordingly one or more encryption and decryption processes corresponding to different encryption layers as previously specified. For instance, network node 906 might further encrypt the message (encrypted data and symmetric key) sent by sender 901 and, on the other side, network node 909 might decrypt the encrypted data and or the encrypted key according to one or more encryption layers before passing it to the recipient 902.

According to another embodiment of the present invention, this layered encryption concept is used to control the data flow through the network, for instance, by the sender 901 or by other nodes of the network 900. With reference to Figures 10 to 12, a high level flow chart is illustrated demonstrating the basic functionality of this process. Referring to Figure 10, exemplary actions as performed by the sender 901 are illustrated whereby the sender is assumed to be already in possession of the public key associated with the intended recipient 902. Following the concept of the layered encryption, the sender encrypts the digital data by using a symmetric key S. Because this symmetric key S and the respective encryption scheme can be seen as a generic encryption scheme that is independent of the concept regarding present invention, a method or system for obtaining the symmetric key is not discussed in greater detail. In general, any encryption scheme could be employed for this purpose.

For the illustrated example, it is assumed that the encrypted data is intended to be sent via node A (906) and node D (909) to recipient 902. Such a path specification via known and predetermined network nodes might for instance be required for security purposes, for the sake of session tracking and logging of information related to the data distribution, which is accomplished by those specified network nodes. It has again been noted that in general any terminal or server connected to that network 900 can be seen as a network node itself and therefore the transmitted data can be required to be sent to a specific authorization or confirmation server prior to being sent to the final recipient 902. Shown as step 111 in Figure 10, the sender

first encrypts the digital data by means of the encryption key S. Then this symmetric key S is itself encrypted using the public key corresponding to the recipient 902 as illustrated in step 112. Following as successive encryption steps 113 and 114, the resultant encrypted symmetric key S is again encrypted applying first the public key PK_d of network node D and second the public key PK_a of network node A in respective encryption processes. This encrypted encryption key S is then sent together with the encrypted data (step 115) to the network, in particular to network node A.

This layered encryption using first the public key of network node D and then the public key of network node A, the sender ensures that the encrypted message is sent via node A and node D to recipient 902, as will be apparent by subsequent descriptions regarding Figures 11 and 12.

Referring to both Figures 10 and 11, the illustrated flow chart as shown in Figure 10 respectively continues as shown in Figure 11.

Network node A receives the message that was sent by the sender 901, whereby the message is comprised of the encrypted digital data and the asymmetrically encrypted key S. Network node A now decrypts the encrypted key S using its private key SK_a, as shown by step 121. Because only network node A is in possession of that private key that corresponds to the public key used by the sender in step 114, only node A can perform this decryption step 121. This decryption step 121 obviously reverses the encryption step 113 that was performed by the sender 901. In order to finally obtain the encryption key S as required to decrypt the encrypted data, all encryption steps must be reversed by a corresponding decryption step. Consequently, the recipient 902, when receiving the message sent by sender 901, can only decrypt the encryption key S if node A has indeed decrypted the encrypted key S respectively. This ensures that the data can only be decrypted if it has passed the network node A. It should further be noted that it is essential for the system security that sender 901 can be certain to have applied an authentic public key PK_a associated with network node A when performing encryption step 114. This required authenticated correspondence of public key PK_a

and network node A is preferably accomplished by distributing public keys by means of the described hash value lists and possibly by digital certificates. The same applies to all public keys employed during this distribution process.

With reference to Figures 9 and 11, node A is assumed to transmit the message to node D, as specified by sender 901. This path specification can be comprised in the message and analyzed by each network node. The message is however send to node D via network node B according to this exemplary illustration. As the flow chart of Figure 11 illustrates, network node A therefore specifies and ensures that the message is sent via node B. Accordingly, the outcome of decryption step 121, the key S which is still encrypted with regard to public key PK_d, is encrypted applying public key of network node B. As discussed for network node A, this ensures that the message has to be sent via node B in order to be able to finally obtain the plain text of the encryption key S. Network node A then sends the encrypted data, together with the encrypted key S, to network node B.

Network node B may now send the received message to still further network nodes other than the anticipated node D (not shown). Required steps would be similar to the steps 120 to 123 of Figure 11. It should, however, be mentioned that step 122 is an optional step and node A might send the message to node B without encrypting the key S especially for network node B.

The exemplary flow chart as shown in Figure 11 continues as shown in Figure 12, whereby network node B receives the message in step 130. The following step 131 corresponds to steps 121 but with respect to node B and its private key. Network node B eventually sends the resultant message to node D which performs a decryption process (step 134) with respect to the encrypted key S and its private key. Finally, the message is sent to recipient 902, shown as step 135. Having received both message parts (step 136), the recipient is now able to decrypt the key S applying its private key PK_R in step 137. With this access to the key S as plain text, the recipient can then decrypt the received data as shown in step 138.

The described distribution process only accounts for the necessary steps as required to provide the person skilled in the art an understanding of the present invention. In general the described process may comprise more steps that account

for other common operations in connection with data distribution via such a network and when applying public key encryption schemes which known by the art. Explicitly mentioned in this sense are optional establishing steps in which a network node determines whether it has to decrypt the key information part and/or the data information part of a received message. Additional steps might determine which further nodes (including the recipient) are specified for that message to be sent to, as well as whether the key information or the message has to be further encrypted with regard to any further node the message is intended to pass. Possible further steps include adding or associating information to the message or parts thereof that specify above routing and encryption specifications.

According to a further embodiment of the present invention, afore described considerations regarding the layered encryption aspects also apply the public key signature schemes. In comparison to the described encryption schemes a public key based digital signature scheme only swaps the roles of all corresponding encryption and decryption processes. Whereby the previous encryption process corresponds to the verification step performed by the recipients and the respective decryption step as performed by the signer or sender of a digital data comprises the operations of the decryption step. Correspondingly applies the sender or signer of data its own private key and the recipient or signature verifier has to obtain the signers public key. Most used signature schemes additionally require a comparison step as part of the signature verification process of the computed and an expected result. Due to the same underlying public key system, the layered encryption and the specified and controlled distribution via predetermined network nodes can be applied to digital signatures. Typically when signing digital data, a hash function is applied on that data and the signing operation involving the signer's private key is performed on that computed hash value. In terms of a layered signature scheme, this would translate to signing the same successively at different signature layers with different private keys, and accordingly verifying it in corresponding verifying processes by the recipient(s).

Usually one would generate a signature at one specific layer by hashing the data and the signature that was produced by the previous layer, since both together would form the message to be signed. Similar to the layered encryption, it would not

be necessary to hash this complete message at each layer. Only the previously computed hash value has to be re-signed respectively with the associated private key of this layer, which might further involve some common data formatting or hashing according to the employed digital signature scheme. This would result in a digital signature for each layer that are mutually independent except that they are performed on the same hash value (of the same digital data in question). Therefore, alternatively, instead of the hash value, the actual signature (the signed hash value) of the respectively previous layer can be signed at each layer. Because the signatures of all (but the first) layers are issued not only on the hashed data but also on the signatures of the previous layer, this alternative has the advantage that it can be easier established and controlled which signature has been issued first and thus must be verified last, i.e. which signature corresponds to which signature layer. With this it is also possible, though not required in all instances, instead of signing the signature of the previous layer to hash the previous signature and sign this new hash value, which now accounts not only for the digital data but also for the previous signature and thus leads to the same conclusion as above. These layered signatures can be combined with those embodiments concerning the layered encryption process of the distributed digital data.

This layered signature method can be applied to different network nodes in the same manner as described for the layered encryption process. Each node, which again might be the sender and/or the recipients as shown in Figure 9, that transmits or receives a message comprised of the (possibly encrypted) data and one or more signatures thereon issues a signature and/or has to verify a signature. Similar schemes for predetermined paths through the network might be specified. The recipient can establish whether the message indeed passed all specified nodes by establishing whether those nodes signed the received message accordingly with a valid signature. Additionally, network nodes different to the anticipated recipient of the transmitted data might as well have to verify the validity of one or more signatures previously issued by other nodes the message passed or originated from before transmitting the message to another node or recipient.

Another aspect of present invention is directed to a secured data distribution in a server client environment wherein a server has to deal with a large number of clients

or users. According to current methods and systems for secured data distribution to a large number of clients established by data encryption are computationally extensive and require large storage capabilities at the server side. Moreover, clients have to register at the system or server prior to the encrypted data transmission, which typically involves disclosing private information about the clients in order to record the same at the server side for future identification and authorization of that clients. This necessity to reveal its identity poses a disadvantage of those encrypted distribution system that occurs usually for symmetric and asymmetric encryption schemes. Even though in an asymmetric encryption method the clients may remain anonymous, i.e. asymmetrically encrypting every data message with the respectively obtained public key, the storage and processing requirements for the server can be significantly high. When using symmetric encryption schemes the server has to store and maintain, at least, a list of all symmetric keys for all participating clients. In addition, both parties have to agreed or establish and/or exchange these keys, which requires a secure and confidentially disclosed data exchange and thus an asymmetric or symmetric encryption. This anonymity and efficiency issue is not only a performance and user convenience matter but directly effects the security of the system since private confidential client information as required for and during the intended secure data distribution process has to be transmitted to an stored at the server side. Therefore, the secured data distribution according to one embodiment of the present invention also accounts for the establishing and handling of required encryption related information, in particular of symmetric keys as shared secrets between the server as the sender and the client as the recipient of the distributed data.

The preferred embodiment therefore comprises a method to establish a secure and efficient communication between a server and a client. Figures 13a and 13b exemplarily illustrate a high level flow chart containing steps that are considered appropriate to describe this aspect of the present invention and to reveal the basic functionalities to those skilled in the art. The flow starts in Figure 13a and continues in Figure 13b respectively as shown. The client initially registers at the server side of the network by sending a randomly generated token T to the server. This token can be asymmetrically encrypted using the servers public key applying the afore-described secure distribution methods as well as the hash value list to distribute the

respective public keys. These steps are illustrated as steps 140 and 141 in Figure 13a.

The server generates or obtains a secure and confidential random value R. This step 142 of Figure 13a can be performed once and then fixed fro this server or can be re-generated or updated over time. Having received and if required decrypted the token T, the server computes a hash value on a message at least comprising the token T and its randomly chosen but fixed value R as shown in step 144. The computed hash value will be denoted as S and serves as symmetric key information for a secure communication between the server and client and accomplishes a shared secret between both. The server provides this shared secret S to the client, which is possibly asymmetrically encrypted using the previously obtained public key of the client as shown by step 145. The server does not have to store this value S or the token S nor the identity of the corresponding client.

In return as illustrated by steps 146 and 147, the client can symmetrically encrypt a message, i.e. digital data, which is to be sent securely and confidentially to the server. Together with this message in question, the token T is also transmitted to the server. For this and for the subsequent message and data transmissions all several alternative embodiments regarding the above described distribution methods can be applied. If the token was encrypted fro instance with the server's public key, the server respectively decrypts the token T as indicated by the optional step 148. The server is then able to easily re-compute the shared secret S that was used to symmetrically the message in question. Identical to step 144, the server hashes the token T and its private random value R in step 149. With this computed value S the server decrypts the received message in step 150. A possibly desired reply message to the client can be encrypted using the same value S and the corresponding symmetric encryption scheme as indicated by steps 151 and 152 because both parties are in possession of the value S that is a secretly shared between them. Alternatively a similar process could be performed using a new, shared secret S.

This method as outlined therefore provides for a symmetric encryption functionality without having to store a list of keys and independent fro all clients that communicate with the server and further provides for the possibility that the clients

can participate anonymously in the system. The security of the system relies on the confidentiality and secrecy of the private random server value R, since this value exclusively provides for a computation of the shared secret S. In case the server uses a new random value R, possibly in compliance of a timed or usage schedule according to the security specification of the system, every client has to register again at the server side, i.e. obtaining the shared secret and key information S in performing steps 140 or 142 to 145.

According to another embodiment, the server generates and/or provides the token T and distributed the same together with the value S to the client. According to further embodiment, the server generates a predetermined part of the token T. With this, the server can compute and distinguish between several random R values, whereby the server determines by means of this generated part of a token T which value R to apply for the computation of the shared secret S. The server might partition the client in different groups and/or might distinguish according to the time of registering of clients at the server. Following the latter concept, the server can control which group of users has to re-register with the system by selectively replacing or updating the respective known part of the token T.

According to yet another embodiment, the token T is replaced by a new token T with every or after a specified number of commutations between server and client. For this, a part of the message that was sent by the client to the server can server as the new token T. Alternatively, it may be used to generate the new token, for instance applying a specified function possibly involving hashing this part of the message. This implies that the server provides the new value S to the client, for instance by sending the same together with a reply message to the client. Particularly, during this communication, the server receives messages from a client. A part of these messages is bases for the new token. According to several considered procedures of this embodiment, this part may be part of the encrypted message, part of the decrypted data of the message, a hash value of one or both of the aforementioned parts or only parts thereof, or any combination of these data sections. The server then computes the new key information S from the new token and the random value R as described and illustrated by steps 144 and 149. The reply message to the client or its data, however, is encrypted using the old previous

key value S as described for Figures 13a and 13b. This encrypted reply message further comprises the new key value S, which is also encrypted using the old S value and is sent together with the reply message to the client. It also possible that the reply message only comprises this new key S.

The client is then able to decrypt the reply message using the old secret share value S according to the respective underlying encryption scheme. The scheme or algorithm fro obtaining the new token T as applied by the server prior to computing the new value S is also known to the client. This might be part of a mutually agreed data transmission protocol. Therefore and because the new token was based on the previous message sent by this very client, the client can compute the new token T in the same manner as the server.

With this method, a new token can be used for every communication between the server and client without the need for a re-registration step of the respective or all clients and without adding computational cost of significance. In addition, if the new token T, and thus the new key information S, is based on at least a portion of the decrypted data of the received message from the client, the it is ensured that no party but the client and the server are able to determine or trace which client was associated with the old token T and is now associated with the new token T.

Consequently with this embodiment, the random value(s) R at the server side can be revoked and replaced without the explicit need for clients to register again at the server, since the new S value is obtained from the new token T and a new random value R. The server therefore obtains a part of the new token not from the received message as described, but generates this part independently thereof and sends this part together and likewise with the new value S to the client. This is important because the server recognizes whether a old or a new random value R has been used and consequently must be re-applied when performing step 149. It is however alternatively also possible to attach some other suitable identifier to the messages for this determination of the correspondingly valid value R, for instance a time specification. Another alternative is a trial and error approach with all possible values R at the server side since this means that low complexity operations have to be performed. Alternatively, the complete token T could be generated by the server and then provides to the client as described above.